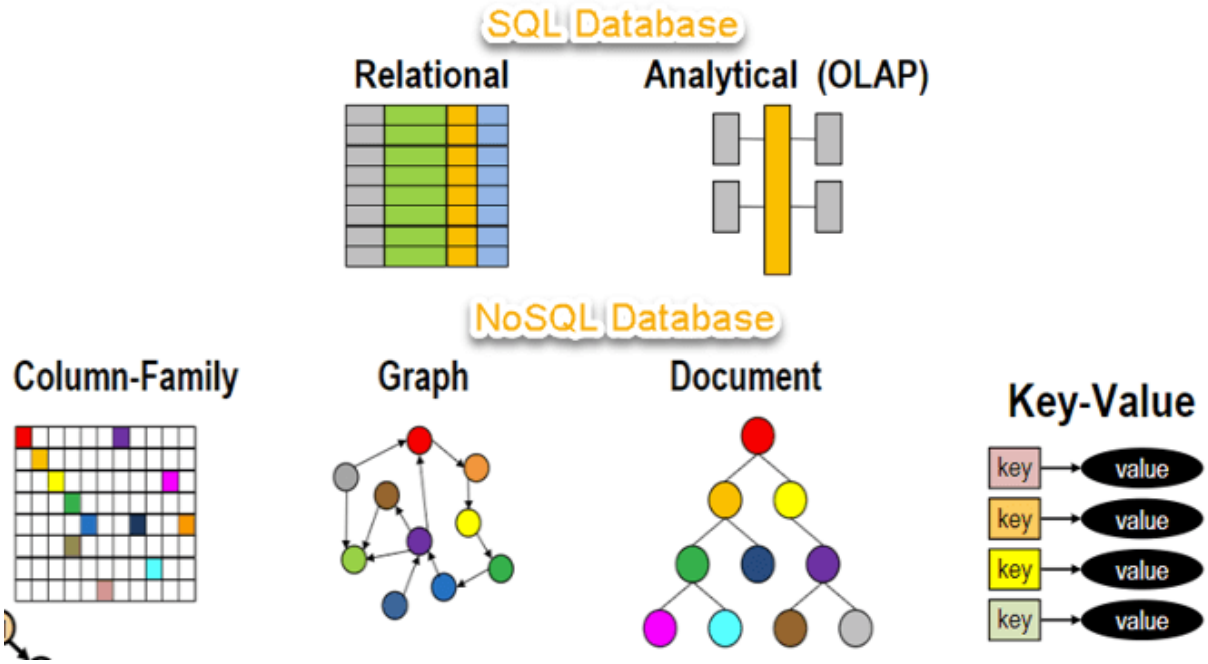
Date⇒ 28-01-2021
Module⇒ Backend
Lecture By⇒ Akash Handa
Subject ⇒ Mongodb Basics

IN PREVIOUS LECTURE **(QUICK RECAP) Date-27/1/2021**	**In Today's Lecture (Overview)**
Request in Node.js Database Collection Document Sql vs Nosql How to Install Mongodb	What Is NoSql Database What is Mongodb How To Setup Mongodb Mongodb Queries/Commands

What Is NoSql Database

When people use the term "NoSQL database", they typically use it to refer to any non-relational database. Some say the term "NoSQL" stands for "non SQL" while others say it stands for "not only SQL." Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

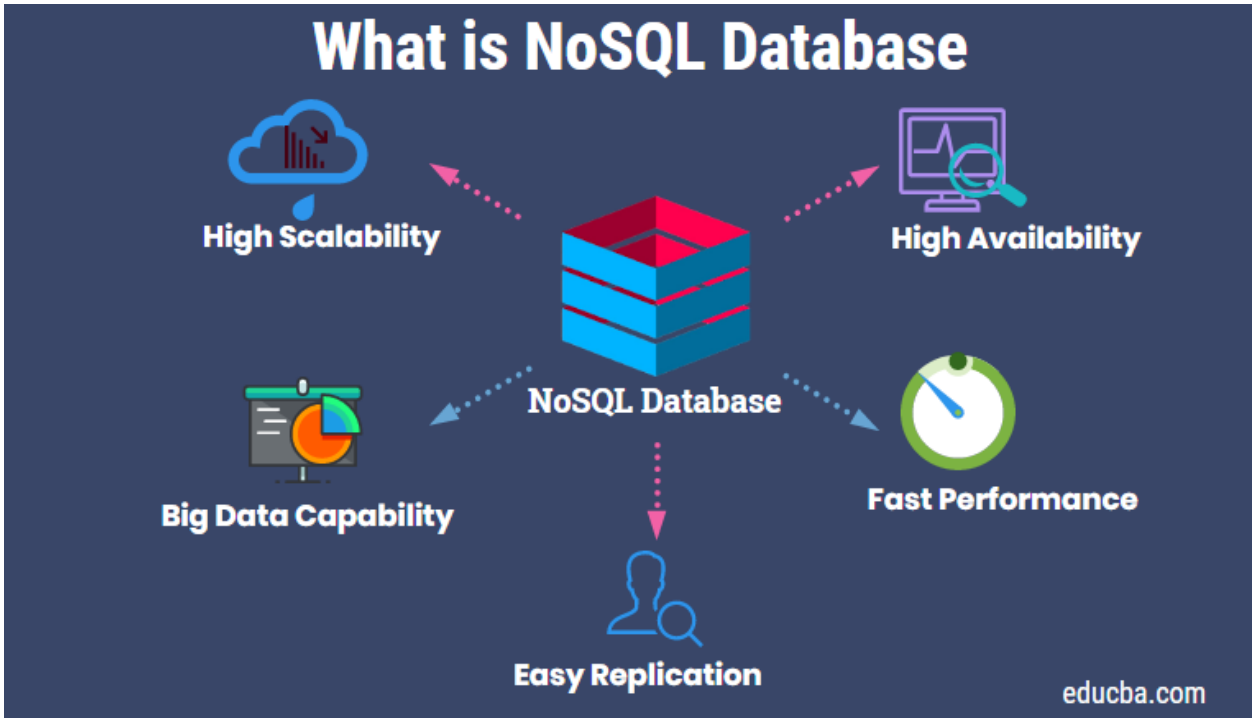
A common misconception is that NoSQL databases or non-relational databases don't store relationship data well. NoSQL databases can store relationship data—they just store it differently than relational databases do. In fact, when compared with SQL databases, many find modeling relationship data in NoSQL databases to be *easier* than in SQL databases, because related data doesn't have to be split between tables.



NoSQL data models allow related data to be nested within a single data structure.

NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model simply for the purposes of reducing data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.

As storage costs rapidly decreased, the amount of data applications needed to store and query increased. This data came in all shapes and sizes—structured, semistructured, and polymorphic—and defining the schema in advance became nearly impossible. NoSQL databases allow developers to store huge amounts of unstructured data, giving them a lot of flexibility.

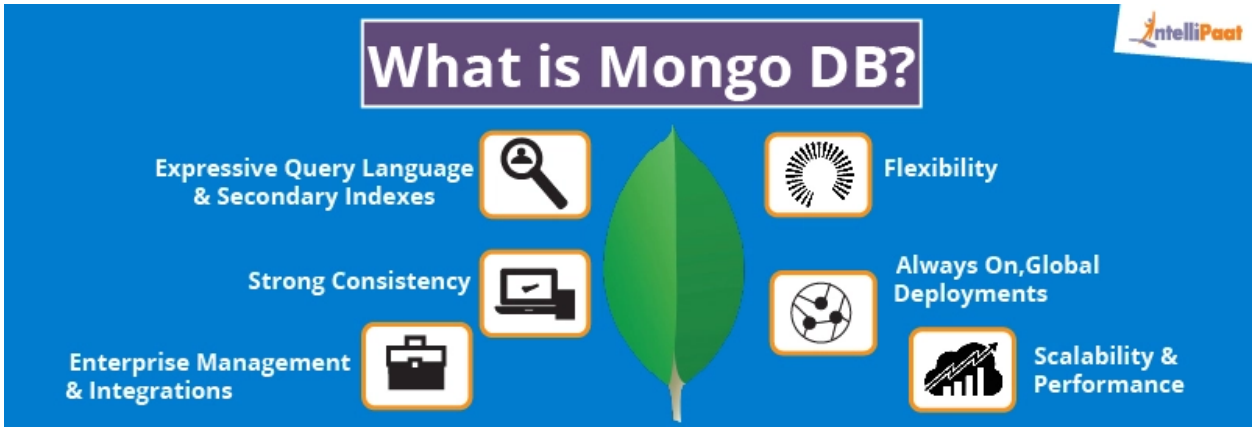


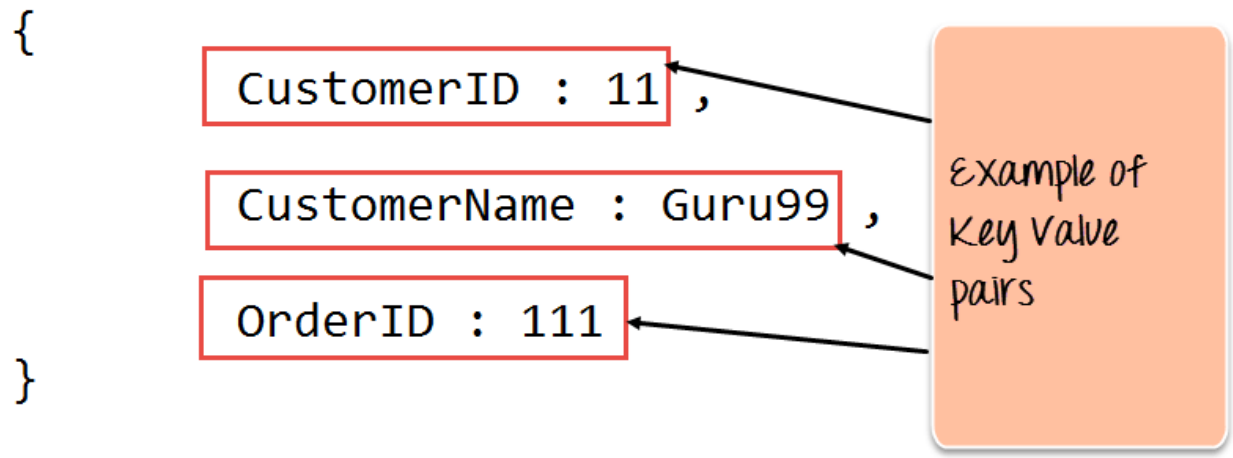
Additionally, the Agile Manifesto was rising in popularity, and software engineers were rethinking the way they developed software. They were recognizing the need to rapidly adapt to changing requirements. They needed the ability to iterate quickly and make changes throughout their software stack—all the way down to the database model. NoSQL databases gave them this flexibility.

Cloud computing also rose in popularity, and developers began using public clouds to host their applications and data. They wanted the ability to distribute data across multiple servers and regions to make their applications resilient, to scale-out instead of scale-up, and to intelligent geo-place their data. Some NoSQL databases like MongoDB provided these capabilities.

What is Mongodb

MongoDB is an object-oriented, simple, dynamic, and scalable NoSQL database. It is based on the NoSQL document store model. The data objects are stored as separate documents inside a collection — instead of storing the data into the columns and rows of a traditional relational database.

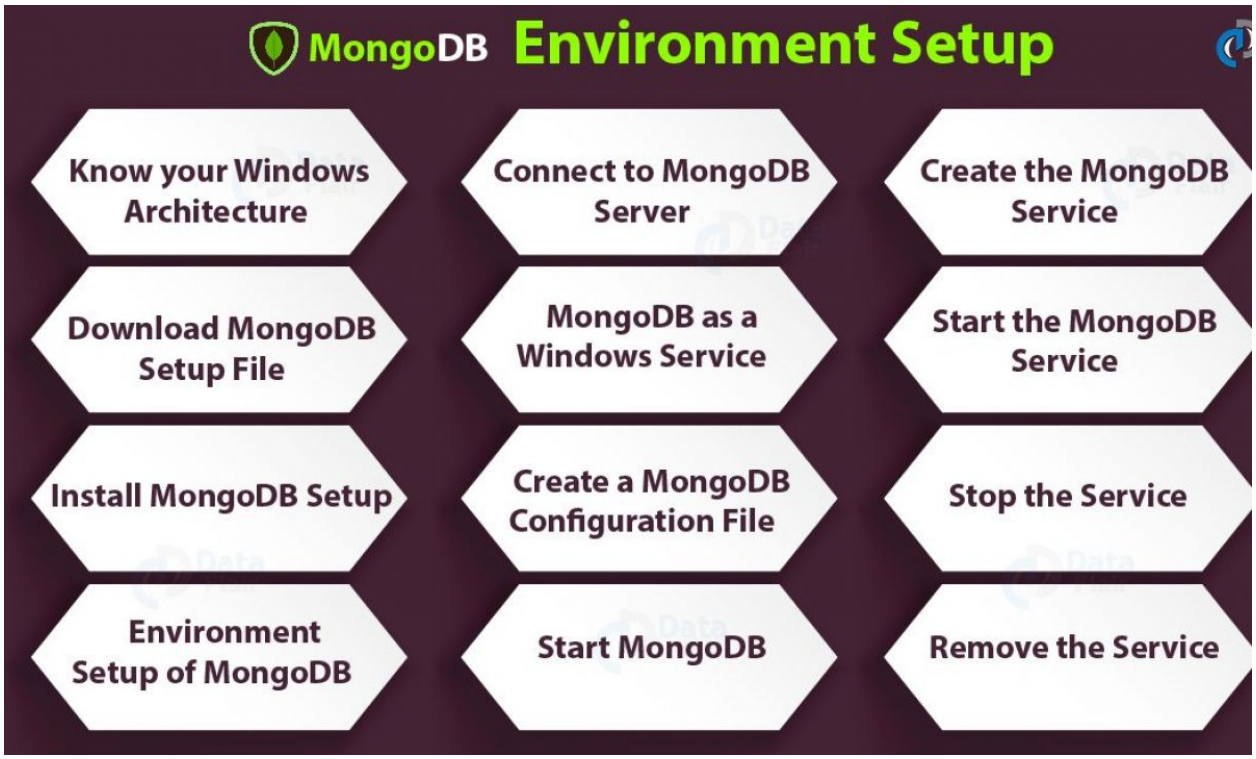




How To Setup Mongodb

First Install Mongodb From the link Given below
https://www.mongodb.com/try/download/community

After Installing it Follow The steps Given below
1.Go to ‘C’ drive in my computer
2.Create folder Called ‘data’
3.In the data folder create db
4.in command line first write mongod
5.and then write mongo to connect your database



Mongodb Queries/Commands

> show dbs	display a list of current databases
> use <database name>	select a database to use. This command will create a new database with the given name if it does not exist.
> db.collection.insertOne()	insert a single item into the given collection
> db.collection.insertMany()	insert multiple items into collection
> db.collection.find()	view all items in a collection

basic mongo commands

* use learn <--- switched to db learn
* > db.getCollectionNames()
* [ ]
* > db.starwars.insert({name: 'C-3PO', gender: 'robot', position: 'Protocol droid', homeworld: 'Tatooine'})
* > db.getCollectionNames()
* ["starwars", "system.indexes"]
* What you're seeing is the name of the index, the database and collection it was created against and the fields included in the index

About this Cheat Sheet

The idea behind this is to have all (well, most) information from the above mentioned Tutorial immediately available in a very compact format. All commands can be used on a small data basis created in the insert-section. All information in this sheet comes **without the slightest warranty for correctness**. Use at your own risk. Have fun ☺!

Basic Information

Download MongoDB	http://www.mongodb.org/downloads
JSON Specification	http://www.json.org/
BSON Specification	http://bsonspec.org/
Java Tutorial	http://www.mongodb.org/display/DOCS/Java+Tutorial

Inserting Documents

```
db.ships.insert({name:'USS Enterprise-D',operator:'Starfleet',type:'Explorer',class:'Galaxy',crew:750,codes:[10,11,12]})
db.ships.insert({name:'USS Prometheus',operator:'Starfleet',class:'Prometheus',crew:4,codes:[1,14,17]})
db.ships.insert({name:'USS Defiant',operator:'Starfleet',class:'Defiant',crew:50,codes:[10,17,19]})
db.ships.insert({name:'IKS Buruk',operator:' Klingon Empire',class:'Warship',crew:40,codes:[100,110,120]})
db.ships.insert({name:'IKS Somraw',operator:' Klingon Empire',class:'Raptor',crew:50,codes:[101,111,120]})
db.ships.insert({name:'Scimitar',operator:'Romulan Star Empire',type:'Warbird',class:'Warbird',crew:25,codes:[201,211,220]})
db.ships.insert({name:'Narada',operator:'Romulan Star Empire',type:'Warbird',class:'Warbird',crew:65,codes:[251,251,220]})
```

Finding Documents

`db.ships.findOne()`	Finds one arbitrary document
`db.ships.find().prettyPrint()`	Finds all documents and using nice formatting
`db.ships.find({}, {name:true, _id:false})`	Shows only the names of the ships
`db.ships.findOne({'name':'USS Defiant'})`	Finds one document by attribute

Basic Concepts & Shell Commands

`db.ships.<command>`	`db` – implicit handle to the used database `ships` – name of the used collection
`use <database>`	Switch to another database
`show collections`	Lists the available collections
`help`	Prints available commands and help

Finding Documents using Operators

`$gt / $gte`	greater than / greater than equals	`db.ships.find({class:{$gt:'P'}`
`$lt / $lte`	lesser than / lesser than equals	`db.ships.find({class:{$lte:'P'}`
`$exists`	does an attribute exist or not	`db.ships.find({type:{$exists:true}})`
`$regex`	Perl-style pattern matching	`db.ships.find({name:{$regex:'^USS\\sE'}})`
`$type`	search by type of an element	`db.ships.find({name : {$type:2}})`

BSON Types

String	2
Array	4
Binary Data	5
Date	9

http://www.w3resource.com/mongodb/mongodb-type-operators.php